The 2018 ACM-ICPC Asia Qingdao Regional Contest, Online

September 16, 2018

Problem List

A	Live Love
B	Red Black Tree
C	Halting Problem
D	Pixel Art
E	Infinite Parenthesis Sequence
F	Chaleur
G	Couleur
H	Traveling on the Axis
I	Kuririn MIRACLE
J	Press the Button
K	XOR Clique

This problem set should contain 11 (eleven) problems on 16 (sixteen) numbered pages.

Prepared by Zhejiang University ACM/ICPC Problem Setter Team.

Problem A. Live Love

DreamGrid is playing the music game *Live Love*. He has just finished a song consisting of n notes and got a result sequence $A_1, A_2, \ldots, A_n$ ($A_i \in \{\text{PERFECT, NON-PERFECT}\}$). The score of the song is equal to the *max-combo* of the result sequence, which is defined as the maximum number of continuous PERFECTs in the sequence.

Formally speaking, max-combo(A) = max $\{k \mid k$ is an integer and there exists an integer i $(1 \le i \le n-k+1)$ such that $A_i = A_{i+1} = A_{i+2} = \cdots = A_{i+k-1} = \text{PERFECT}\}$. For completeness, we define $\max(\emptyset) = 0$.

As DreamGrid is forgetful, he forgets the result sequence immediately after finishing the song. All he knows is the sequence length n and the total number of PERFECTs in the sequence, indicated by m. Any possible score s he may get must satisfy that there exists a sequence A' of length n containing exactly m PERFECTs and $(n-m)$ NON-PERFECTs and max-combo$(A') = s$. Now he needs your help to find the maximum and minimum s among all possible scores.

Input

There are multiple test cases. The first line of the input contains an integer $T(1 \le T \le 100)$, indicating the number of test cases. For each test case:

The only line contains two integers n and m ($1 \le n \le 10^3$, $0 \le m \le 10^3$, $m \le n$), indicating the sequence length and the number of PERFECTs DreamGrid gets.

Output

For each test case output one line containing two integers s_{max} and s_{min}, indicating the maximum and minimum possible score.

Example

standard input	standard output
5 5 4 100 50 252 52 3 0 10 10	4 2 50 1 52 1 0 0 10 10

Note

Let's indicate a PERFECT as P and a NON-PERFECT as N.

For the first sample test case, the sequence (P, P, P, P, N) leads to the maximum score and the sequence (P, P, N, P, P) leads to the minimum score.

Problem B. Red Black Tree

BaoBao has just found a rooted tree with n vertices and $(n-1)$ weighted edges in his backyard. Among the vertices, m of them are red, while the others are black. The root of the tree is vertex 1 and it's a red vertex.

Let's define the cost of a red vertex to be 0, and the cost of a black vertex to be the distance between this vertex and its nearest red ancestor.

Recall that

- The length of a path on the tree is the sum of the weights of the edges in this path.
- The distance between two vertices is the length of the shortest path on the tree to go from one vertex to the other.
- Vertex u is the ancestor of vertex v if it lies on the shortest path between vertex v and the root of the tree (which is vertex 1 in this problem).

As BaoBao is bored, he decides to play q games with the tree. For the i-th game, BaoBao will select k_i vertices $v_{i,1}, v_{i,2}, \ldots, v_{i,k_i}$ on the tree and try to minimize the maximum cost of these k_i vertices by changing at most one vertex on the tree to a red vertex.

Note that

- BaoBao is free to change any vertex among all the n vertices to a red vertex, NOT necessary among the k_i vertiecs whose maximum cost he tries to minimize.
- All the q games are independent. That is to say, the tree BaoBao plays with in each game is always the initial given tree, NOT the tree modified during the last game by changing at most one vertex.

Please help BaoBao calculate the smallest possible maximum cost of the given k_i vertices in each game after changing at most one vertex to a red vertex.

Input

There are multiple test cases. The first line of the input is an integer T, indicating the number of test cases. For each test case:

The first line contains three integers n, m and q ($2 \le m \le n \le 10^5$, $1 \le q \le 2 \times 10^5$), indicating the size of the tree, the number of red vertices and the number of games.

The second line contains m integers $r_1, r_2, \ldots, r_m$ ($1 = r_1 < r_2 < \cdots < r_m \le n$), indicating the red vertices.

The following $(n-1)$ lines each contains three integers u_i, v_i and w_i ($1 \le u_i, v_i \le n$, $1 \le w_i \le 10^9$), indicating an edge with weight w_i connecting vertex u_i and v_i in the tree.

For the following q lines, the i-th line will first contain an integer k_i ($1 \le k_i \le n$). Then k_i integers $v_{i,1}, v_{i,2}, \ldots, v_{i,k_i}$ follow ($1 \le v_{i,1} < v_{i,2} < \cdots < v_{i,k_i} \le n$), indicating the vertices whose maximum cost BaoBao has to minimize.

It's guaranteed that the sum of n in all test cases will not exceed 10^6, and the sum of k_i in all test cases will not exceed 2×10^6.

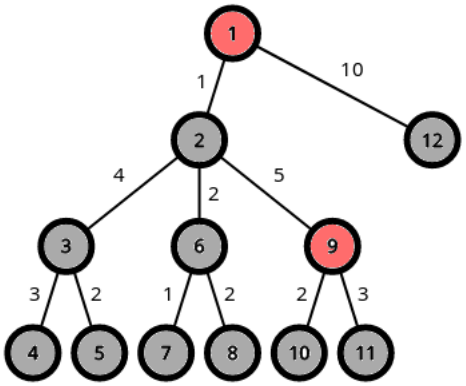
Output

For each test case output q lines each containing one integer, indicating the smallest possible maximum cost of the k_i vertices given in each game after changing at most one vertex in the tree to a red vertex.

Example

standard input	standard output
2 12 2 4 1 9 1 2 1 2 3 4 3 4 3 3 5 2 2 6 2 6 7 1 6 8 2 2 9 5 9 10 2 9 11 3 1 12 10 3 3 7 8 4 4 5 7 8 4 7 8 10 11 3 4 5 12 3 2 3 1 2 1 2 1 1 3 1 1 1 2 1 2 3 1 2 3	4 5 3 8 0 0 0

Note



The first sample test case is shown above. Let's denote $C(v)$ as the cost of vertex v.

For the 1st game, the best choice is to make vertex 2 red, so that $C(3) = 4$, $C(7) = 3$ and $C(8) = 4$. So the answer is 4.

For the 2nd game, the best choice is to make vertex 3 red, so that $C(4) = 3$, $C(5) = 2$, $C(7) = 4$ and $C(8) = 5$. So the answer is 5.

For the 3rd game, the best choice is to make vertex 6 red, so that $C(7) = 1$, $C(8) = 2$, $C(10) = 2$ and $C(11) = 3$. So the answer is 3.

For the 4th game, the best choice is to make vertex 12 red, so that $C(4) = 8$, $C(5) = 7$ and $C(12) = 0$. So the answer is 8.

Problem C. Halting Problem

In computability theory, the halting problem is the problem of determining, from a description of an arbitrary computer program, whether the program will finish running (i.e., halt) or continue to run forever.

Alan Turing proved in 1936 that a general algorithm to solve the halting problem cannot exist, but DreamGrid, our beloved algorithm scientist, declares that he has just found a solution to the halting problem in a specific programming language – the Dream Language!

Dream Language is a programming language consists of only 5 types of instructions. All these instructions will read from or write to a 8-bit register r, whose value is initially set to 0. We now present the 5 types of instructions in the following table. Note that we denote the current instruction as the i-th instruction.

Instruction	Description
add v	Add v to the register r. As r is a 8-bit register, this instruction actually calculates $(r + v) \mod 256$ and stores the result into r, i.e. $r \leftarrow (r + v) \mod 256$. After that, go on to the $(i + 1)$-th instruction.
beq v k	If the value of r is equal to v, jump to the k-th instruction, otherwise go on to the $(i + 1)$-th instruction.
bne v k	If the value of r isn't equal to v, jump to the k-th instruction, otherwise go on to the $(i + 1)$-th instruction.
blt v k	If the value of r is strictly smaller than v, jump to the k-th instruction, otherwise go on to the $(i + 1)$-th instruction.
bgt v k	If the value of r is strictly larger than v, jump to the k-th instruction, otherwise go on to the $(i + 1)$-th instruction.

A Dream Language program consisting of n instructions will always start executing from the 1st instruction, and will only halt (that is to say, stop executing) when the program tries to go on to the $(n + 1)$-th instruction.

As DreamGrid's assistant, in order to help him win the Turing Award, you are asked to write a program to determine whether a given Dream Language program will eventually halt or not.

Input

There are multiple test cases. The first line of the input is an integer T, indicating the number of test cases. For each test case:

The first line contains an integer n ($1 \le n \le 10^4$), indicating the number of instructions in the following Dream Language program.

For the following n lines, the i-th line first contains a string s ($s \in$ {"add","beq","bne","blt","bgt"}), indicating the type of the i-th instruction of the program.

- If s equals to "add", an integer v follows ($0 \le v \le 255$), indicating the value added to the register;
- Otherwise, two integers v and k follow ($0 \le v \le 255$, $1 \le k \le n$), indicating the condition value and the destination of the jump.

It's guaranteed that the sum of n of all test cases will not exceed 10^5.

Output

For each test case output one line. If the program will eventually halt, output "Yes" (without quotes); If the program will continue to run forever, output "No" (without quotes).

Example

standard input	standard output
4 2 add 1 blt 5 1 3 add 252 add 1 bgt 252 2 2 add 2 bne 7 1 3 add 1 bne 252 1 beq 252 1	Yes Yes No No

Note

For the second sample test case, note that r is a 8-bit register, so after four "add 1" instructions the value of r will change from 252 to 0, and the program will halt.

For the third sample test case, it's easy to discover that the value of r will always be even, so it's impossible for the value of r to be equal to 7, and the program will run forever.

Problem D. Pixel Art

DreamGrid is creating a piece of pixel art on a grid of n rows and m columns where all the cells are initially white. As a beginner pixel artist, he can only paint horizontal or vertical lines on the grid.

Let's denote the cell on the i-th row and the j-th column as (i, j). Formally speaking,

- A horizontal line (r, c_1, c_2) where $c_1 \leq c_2$ consists of all the cells (i, j) satisfying $i = r$ and $c_1 \leq j \leq c_2$;
- A vertical line (c, r_1, r_2) where $r_1 \leq r_2$ consists of all the cells (i, j) satisfying $j = c$ and $r_1 \leq i \leq r_2$;
- By painting a line on the grid, DreamGrid will turn all the cells in that line into black cells.

After some time, DreamGrid has left k non-intersecting lines on the grid. As DreamGrid is also an enthusiast for competitive programming, he is interested in the mathematical properties of his artwork as well.

Let G_i be the sub-grid containing the the first i rows of the original $n \times m$ grid. We denote s_i as the number of black cells in G_i and c_i as the number of black connected components in G_i. Can you help DreamGrid calculate the value of s_i and c_i for all $1 \leq i \leq n$, so that he can have a deeper insight into his artwork?

Note that in this problem, two black cells (r_a, c_a) and (r_b, c_b) are in the same connected component, if there exists a sequence of black cells $(r_1, c_1), (r_2, c_2), \ldots, (r_k, c_k)$ such that $r_1 = r_a, c_1 = c_a, r_k = r_b, c_k = c_b$ and for all $1 \leq i < k$, (r_i, c_i) and (r_{i+1}, c_{i+1}) share an edge.

Input

There are multiple test cases. The first line of the input contains an integer T, indicating the number of test cases. For each test case:

The first line contains three integers n, m and k ($1 \leq n, m, k \leq 10^5$), indicating the number of rows and the number of columns of the grid, and the number of lines DreamGrid has painted.

The following k lines each contains four integers r_1, c_1, r_2, c_2 ($1 \leq r_1 \leq r_2 \leq n$, $1 \leq c_1 \leq c_2 \leq m$, $r_1 = r_2$ or $c_1 = c_2$) indicating the lines DreamGrid painted on the grid.

- $r_1 = r_2$ indicates that DreamGrid has painted a horizontal line (r_1, c_1, c_2);
- $c_1 = c_2$ indicates that DreamGrid has painted a vertical line (c_1, r_1, r_2);
- It's possible that both $r_1 = r_2$ and $c_1 = c_2$ hold, indicating that DreamGrid has turned the cell (r_1, c_1) to a black cell.

It's guaranteed that

- No two lines in the same test case will intersect. Two lines intersect if there exists a cell which is contained in both lines;
- Neither the sum of n nor the sum of k of all test cases will exceed 5×10^5. Note that there is no guarantee on the sum of m.

Output

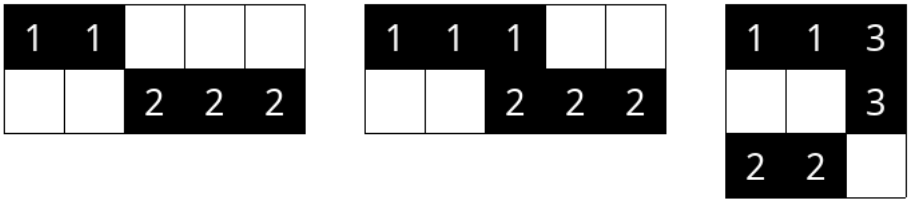
For each test case output n lines where the i-th line contains two integers s_i and c_i separated by a space, indicating the number of black cells and the number of black connected components in the first i rows of the grid.

Example

standard input	standard output
3 2 5 2 1 1 1 2 2 3 2 5 2 5 2 1 1 1 3 2 3 2 5 3 3 3 1 1 1 2 3 1 3 2 1 3 2 3	2 1 5 2 3 1 6 1 3 1 4 1 6 2

Note

The following image illustrates the sample test cases. The number in the cell indicates the index of the corresponding line.



For the third sample test case, it's obvious that there are 3 black cells and 1 black connected components in the first row, 4 black cells and 1 black connected components in the first two rows, and 6 black cells and 2 black connected components in all the three rows. So the answer is "3 1", "4 1" and "6 2".

Problem E. Infinite Parenthesis Sequence

BaoBao has just found a sequence $A = a_0, a_1, \ldots, a_{n-1}$ of length n in his left pocket. Each element a_i in this sequence is either a left parenthesis '(' or a right parenthesis ')'. As BaoBao dislikes short sequences, he decides to make the sequence infinitely long!

Let's denote b_i as the element in the i-th position of the infinite parenthesis sequence B. As B is an infinite sequence, i can be positive, zero, or even negative! To derive B from A, one can use the following equations:

$$\begin{cases} b_i = a_i & \text{if } 0 \le i < n \\ b_i = b_{i-n} & \text{if } i \ge n \\ b_i = b_{i+n} & \text{if } i < 0 \end{cases}$$

As BaoBao is bored, he also crafts a generator to generate an infinite number of parenthesis sequences from sequence B! Denote B^k ($k \ge 1$) as the k-th infinite sequence generated by the generator and b_i^k as the element in the i-th position of sequence B^k. For completeness, we define $B^0 = B$. One can derive B^k from B^{k-1} using the following equations:

$$\begin{cases} b_i^k = b_{i+1}^{k-1} & \text{if } b_i^{k-1} = \text{'('} \\ b_i^k = b_{i-1}^{k-1} & \text{if } b_i^{k-1} = \text{')'} \end{cases}$$

To obtain a deeper insight of the sequence, BaoBao would like to calculate the number of left parenthesis '(' in the continuous subsequence $b_l^k, b_{l+1}^k, b_{l+2}^k, \ldots, b_{r-1}^k, b_r^k$ of B^k. Please write a program to help him calculate the answer.

Input

There are multiple test cases. The first line of the input contains an integer T, indicating the number of test cases. For each test case:

The first line contains a string s ($1 \le |s| \le 10^5$, $s_i \in \{\text{'('}, \text{')'}\}$) indicating the sequence A. The i-th character s_i in s indicates the value of a_{i-1}.

The second line contains an integer q ($1 \le q \le 10^5$), indicating the number of queries.

For the following q lines, each line contains three integers k, l and r ($0 \le k \le 10^9$, $-10^9 \le l \le r \le 10^9$), indicating a query.

It's guaranteed that neither the sum of $|s|$ nor the sum of q of all test cases will exceed 10^6.

Output

For each query output one line containing one integer, indicating the number of left parenthesis '(' in the continuous subsequence $b_l^k, b_{l+1}^k, b_{l+2}^k, \ldots, b_{r-1}^k, b_r^k$ of B^k.

Example

standard input	standard output
3 (()) 3 0 -3 2 1 -2 3 2 0 0))()(3 0 -3 4 2 1 3 3 -4 -1))()((()(4 1234 -5678 9012 123 -456 789 12 -34 56 1 -2 3	3 3 0 4 1 1 7345 623 45 3

Note

In the following explanation, the value of b_0^k is marked in ***bold and italics***.

For the first sample test case, we have $B^0 = ...(())\boldsymbol{(}())(())..., B^1 = ...()()\boldsymbol{(})()()()...$ and $B^2 = ...)()()\boldsymbol{()}()()()(...$, so the answer is 3, 3 and 0.

For the second sample test case, we have $B^0 = ...))()()\boldsymbol{()})()())()(..., B^1 = ...())()\boldsymbol{()})())()()...$, $B^2 = ...)())()\boldsymbol{()})()()()(...$ and $B^3 = ...()())\boldsymbol{()})()()...$, so the answer is 4, 1 and 1.

Problem F. Chaleur

DreamGrid has n friends which are conveniently numbered from 1 to n. They can be divided into two groups (possibly empty) such that:

- Every pair of friends in the first group have to know each other.
- Every pair of friends in the second group must not know each other.

Now, given the pairs of friends who know each other, DreamGrid would like to know the number of ways to find a group of friends with maximum size such that every pair of friends in the group have to know each other, and he would also like to know the number of ways to find a group of friends with maximum size such that every pair of friends in the group must not know each other.

Input

There are multiple test cases. The first line of input contains an integer T, indicating the number of test cases. For each test case:

The first line contains two integers n and m ($1 \leq n \leq 10^5, 0 \leq m \leq 10^5$) – the number of friends and the number of pairs of friends who know each other.

The i-th of the following m lines contains two integers a_i and b_i ($1 \leq a_i, b_i \leq n, a_i \neq b_i$), which denotes that the a_i-th friend and the b_i-th friend know each other. Note that every unordered pair of (a, b) will appear at most once.

It is guaranteed that neither the sum of all n nor the sum of all m exceeds 2×10^6.

Output

For each test case, output two integers separated by a single space.

The first integer indicates the number of ways to find a group of friends with maximum size such that every pair of friends in the this group have to know each other.

The second integer indicates the number of ways to find a group of friends with maximum size such that every pair of friends in the group must not know each other.

Example

standard input	standard output
3 3 2 1 2 2 3 6 6 1 2 2 3 1 3 1 4 2 5 3 6 4 1 1 2	2 1 1 4 1 2

Problem G. Couleur

DreamGrid has an array of n integers. On this array he can perform the following operation: choose an element that was not previously chosen and mark it as unavailable. DreamGrid would like to perform exactly n operations until all the elements are marked.

DreamGrid defines the cost of a subarray as the number of inversions in the subarray. Before performing an operation, DreamGrid would like to know the maximum cost of a subarray that doesn't contain any unavailable elements.

Recall that a subarray $a_l, a_{l+1}, \ldots, a_{r-1}, a_r$ is a **contiguous** subpart of the original array where $1 \le l \le r \le n$. An inversion in a subarray $a_l, a_{l+1}, \ldots, a_{r-1}, a_r$ is a pair of indices (i, j) $(l \le i < j \le r)$ such that the inequality $a_i > a_j$ holds.

Input

There are multiple test cases. The first line of input contains an integer T, indicating the number of test cases. For each test case:

The first line contains a single integer n $(1 \le n \le 10^5)$ – the length of the array.

The second line contains the n values of the array $a_1, a_2, ..., a_n$ $(1 \le a_i \le n)$.

The third line contains a permutation $p_1, p_2, \ldots, p_n$, representing the indices of the elements chosen for the operations in order.

Note that the permutation is encrypted and you can get the real permutation using the following method: Let z_i be the answer before the i-th operation. The actual index of the i-th operation is $p_i \oplus z_i$ where $\oplus$ is bitwise exclusive or operator.

It is guaranteed that the sum of all n does not exceed 10^6.

Output

For each test case, output n integers $z_1, z_2, \ldots, z_n$ in a single line seperated by one space, where z_i is the answer before the i-th operation.

Please, DO NOT output extra spaces at the end of each line, or your answer may be considered incorrect!

Example

standard input	standard output
3 5 4 3 1 1 1 5 4 5 3 1 10 9 7 1 4 7 8 5 7 4 8 21 8 15 5 9 2 4 5 10 6 15 4 8 8 1 12 1 10 14 7 14 2 9 13 10 3 37 19 23 15 7 2 10 15 2 13 4 5 8 7 10	7 0 0 0 0 20 11 7 2 0 0 0 0 0 0 42 31 21 14 14 4 1 1 1 0 0 0 0 0 0

Note

The decoded permutation of each test case is $\{2, 4, 5, 3, 1\}$, $\{1, 3, 8, 7, 9, 2, 4, 5, 10, 6\}$ and $\{15, 12, 2, 1, 9, 6, 11, 14, 3, 13, 4, 5, 8, 7, 10\}$

Problem H. Traveling on the Axis

BaoBao is taking a walk in the interval $[0, n]$ on the number axis, but he is not free to move, as at every point $(i - 0.5)$ for all $i \in [1, n]$, where i is an integer, stands a traffic light of type t_i ($t_i \in \{0, 1\}$).

BaoBao decides to begin his walk from point p and end his walk at point q (both p and q are integers, and $p < q$). During each unit of time, the following events will happen **in order**:

1. Let's say BaoBao is currently at point x, he will then check the traffic light at point $(x + 0.5)$. If the traffic light is green, BaoBao will move to point $(x + 1)$; If the traffic light is red, BaoBao will remain at point x.
2. All the traffic lights change their colors. If a traffic light is currently red, it will change to green; If a traffic light is currently green, it will change to red.

A traffic light of type 0 is initially red, and a traffic light of type 1 is initially green.

Denote $t(p, q)$ as the total units of time BaoBao needs to move from point p to point q. For some reason, BaoBao wants you to help him calculate

$$\sum_{p=0}^{n-1} \sum_{q=p+1}^{n} t(p, q)$$

where both p and q are integers. Can you help him?

Input

There are multiple test cases. The first line of the input contains an integer T, indicating the number of test cases. For each test case:

The first and only line contains a string s ($1 \le |s| \le 10^5$, $|s| = n$, $s_i \in \{\text{'0'}, \text{'1'}\}$ for all $1 \le i \le |s|$), indicating the types of the traffic lights. If $s_i =$ '0', the traffic light at point $(i - 0.5)$ is of type 0 and is initially red; If $s_i =$ '1', the traffic light at point $(i - 0.5)$ is of type 1 and is initially green.

It's guaranteed that the sum of $|s|$ of all test cases will not exceed 10^6.

Output

For each test case output one line containing one integer, indicating the answer.

Example

standard input	standard output
3 101 011 11010	12 15 43

Note

For the first sample test case, it's easy to calculate that $t(0, 1) = 1$, $t(0, 2) = 2$, $t(0, 3) = 3$, $t(1, 2) = 2$, $t(1, 3) = 3$ and $t(2, 3) = 1$, so the answer is $1 + 2 + 3 + 2 + 3 + 1 = 12$.

For the second sample test case, it's easy to calculate that $t(0, 1) = 2$, $t(0, 2) = 3$, $t(0, 3) = 5$, $t(1, 2) = 1$, $t(1, 3) = 3$ and $t(2, 3) = 1$, so the answer is $2 + 3 + 5 + 1 + 3 + 1 = 15$.

Problem I. Kuririn MIRACLE

As we know, DreamGrid is a master of planning algorithm. Recently, a new type of autonomous car appears in Gridland. With the shape of a circle, the car can move in any direction. Being interested in this car, DreamGrid decides to design a specific path planning algorithm for it. Here is the first case he wants to solve.

Consider two cars on the two-dimensional plane with the same radius of r. We now want to move the center of the first car from $(0, 0)$ to $(d, 0)$, where d is positive. The second car, starting its move with its center located at $(2r, 0)$, can be considered as an obstacle moving towards the positive direction of the x-axis with a **constant** speed of v. Luckily, after installing a new engine, the first car can move twice as fast as the obstacle car, which means the **maximum** speed of the first car can be $2v$.

DreamGrid wants to know the shortest time needed to move the first car to the end point without colliding with the second car. That is to say, before arriving at the end point, the circle representing the first car can't intersect with (but can be tangent to) the circle representing the second car.

As DreamGrid is too busy, you are asked to solve this simple problem for him. Of course, if you successfully solve this problem, you will get a brand-new autonomous car in the Gridland!

Input

There are multiple test cases. The first line of the input is an integer $T(1 \leq T \leq 1000)$, indicating the number of test cases. For each test case:

The only line contains three real numbers v, r, d $(1 \leq v, r \leq 10, 1 \leq d \leq 100)$ with at most two digits after the decimal point, indicating the radius of the car, the speed of the obstacle car and the distance between the start point and end point.

Output

For each test case output one line, indicating the shortest time for the first car to arrive the end point without colliding with the second car. Your answer will be considered correct if and only if the absolute error or relative error of your answer is less than 10^{-6}.

Example

standard input	standard output
1 2.00 3 30.0	8.310579933902352

Problem J. Press the Button

BaoBao and DreamGrid are playing a game using a strange button. This button is attached to an LED light (the light is initially off), a counter and a timer and functions as follows:

- When the button is pressed, the timer is set to $(v + 0.5)$ seconds (no matter what the value of the timer is before the button is pressed), where v is a given integer, and starts counting down;
- When the button is pressed with the LED light off, the LED light will be lit up;
- When the button is pressed with the LED light on, the value of the counter will be increased by 1;
- When the timer counts down to 0, the LED light turns off.

During the game, BaoBao and DreamGrid will press the button periodically. If the current real time (that is to say, the time elapsed after the game starts, NOT the value of the timer) in seconds is an integer and is a multiple of a given integer a, BaoBao will immediately press the button b times; If the current time in seconds is an integer and is a multiple of another given integer c, DreamGrid will immediately press the button d times.

Note that

- 0 is a multiple of every integer;
- Both BaoBao and DreamGrid are good at pressing the button, so it takes no time for them to finish pressing;
- If BaoBao and DreamGrid are scheduled to press the button at the same second, DreamGrid will begin pressing the button d times after BaoBao finishes pressing the button b times.

The game starts at 0 second and ends after t seconds (if the button will be pressed at t seconds, the game will end after the button is pressed). What's the value of the counter when the game ends?

Input

There are multiple test cases. The first line of the input contains an integer T (about 100), indicating the number of test cases. For each test case:

The first and only line contains six integers a, b, c, d, v and t ($1 \le a, b, c, d \le 10^6$, $1 \le v, t \le 10^{12}$). Their meanings are described above.

Output

For each test case output one line containing one integer, indicating the value of the counter when the game ends.

Example

standard input	standard output
2 8 2 5 1 2 18 10 2 5 1 2 10	6 4

Note

We now explain the first sample test case.

- At 0 second, the LED light is initially off. After BaoBao presses the button 2 times, the LED light turns on and the value of the counter changes to 1. The value of the timer is also set to 2.5 seconds. After DreamGrid presses the button 1 time, the value of the counter changes to 2.

- At 2.5 seconds, the timer counts down to 0 and the LED light is off.
- At 5 seconds, after DreamGrid presses the button 1 time, the LED light is on, and the value of the timer is set to 2.5 seconds.
- At 7.5 seconds, the timer counts down to 0 and the LED light is off.
- At 8 seconds, after BaoBao presses the button 2 times, the LED light is on, the value of the counter changes to 3, and the value of the timer is set to 2.5 seconds.
- At 10 seconds, after DreamGrid presses the button 1 time, the value of the counter changes to 4, and the value of the timer is changed from 0.5 seconds to 2.5 seconds.
- At 12.5 seconds, the timer counts down to 0 and the LED light is off.
- At 15 seconds, after DreamGrid presses the button 1 time, the LED light is on, and the value of the timer is set to 2.5 seconds.
- At 16 seconds, after BaoBao presses the button 2 times, the value of the counter changes to 6, and the value of the timer is changed from 1.5 seconds to 2.5 seconds.
- At 18 seconds, the game ends.

Problem K. XOR Clique

BaoBao has a sequence $a_1, a_2, \ldots, a_n$. He would like to find a subset S of $\{1, 2, \ldots, n\}$ such that $\forall i, j \in S$, $a_i \oplus a_j < \min(a_i, a_j)$ and $|S|$ is maximum, where $\oplus$ means bitwise exclusive or.

Input

There are multiple test cases. The first line of input contains an integer T, indicating the number of test cases. For each test case:

The first line contains an integer n ($1 \leq n \leq 10^5$), indicating the length of the sequence.

The second line contains n integers: $a_1, a_2, \ldots, a_n$ ($1 \leq a_i \leq 10^9$), indicating the sequence.

It is guaranteed that the sum of n in all cases does not exceed 10^5.

Output

For each test case, output an integer denoting the maximum size of S.

Example

standard input	standard output
3 3 1 2 3 3 1 1 1 5 1 2323 534 534 5	2 3 2